JTRS Document Change Proposal No. 99

Date Initiated: **06/12/2000** ***Status:*** **Approved**

Submitter: Bickle Jerry

Company/ Organization: Raytheon

Document: Software Communications Architecture (SCA) ***Version:*** 1

Reference: (General)

Description: Errata 1 to SCA 1.0

Comment Submitted: There are several errors in the text, or areas which need clarification to properly understand the requirements.

Submitter's Recommen-dation: See Errata 1 to the SCA 1.0

Submitter's Rationale: Clarify current SCA without changing any technical requirements.

Initial Evaluation:

Entry Date: 06/12/2000 ***Evaluation Date:*** 06/12/2000

Priority: ***Category:*** Administrative

Action:

Engineering Action:

Engineering Action Date: 06/12/2000

Proposed Resolution: See attached SCA v1.0 errata 1 file.

Final CCB Action: ***Final Action Date:*** 06/12/2000

Final CCB Resolution: Approved. Post errata sheet with CCB report as cover sheet to JTRS website.
Incorporate changes in next release (v1.x)

CCB Resolution to be incorporated in document version number: v1.x

Date Last Modified: ***06/12/2000***

End of Comment ***99***

Errata 1 to SCA Specification v1.0

6 June 2000

The following changes are provided for clarification of the Software Communications Architecture Specification and will be incorporated in the next release of the document.

Page	Section / Reference	Change
3-3	Figure 3-2	There should be a using relationship to *Device* from *DomainManager* not an aggregate relationship. *StringConsumer* is not implemented by the CF but is used by the CF *Logger*. The uses relationship between *DeviceManager* and *FileManager* should be an aggregate relationship with a cardinality of 1; the aggregate relationship between *DomainManager* and *FileManager* should have a cardinality of 1 (rather than 1..*).
3-4	3.1.3	Delete last paragraph due to status of CORBA extensions and security definition: ~~If user security controls are required, then the CORBA context capability shall be used for setting the user access authorities that are passed along with the CF call. This context information is used by the CF implementation for verifying user authorization before the CF operation is performed. The context content and format is implementation specific. In addition, the file verification mechanism to be used is implementation specific.~~ Note: this requirement will be reconsidered as part of the continuing Security Architecture definition.
3-6	3.1.3.1.2.3.1	Correct *LifeCycle* exception text to match IDL: InitializeError. `exception InitializeError { ~~string~~ StringSequence error~~m~~Message; };`
3-6	3.1.3.1.2.3.2	Correct *LifeCycle* exception text to match IDL: ReleaseError. `exception ReleaseError { ~~s~~StringSequence error~~m~~Message; };`
3-14	3.1.3.1.6.5.1.1	Clarification: ~~Applications may need to create *Resources* in another address space (e.g., process space, another processor, etc.) or without having the ability to directly create the *Resource* servant (e.g. the servant may be provided as part of the implementation of a commercial library).~~ This operation provides the capability to create *Resources* in the same process space as the *ResourceFactory* or to return a *Resource* that has already been created. This behavior is an alternative approach for creating a *Resource* to the *Device execute* operations.
3-23	3.1.3.2.2.5.1.3	correction to step 3: … needed by the *Application*. Create an ~~ApplicationProfile~~ instance of an *Application,* if the requested *Application* can be created. Update the *Device*(s) …
3-23	3.1.3.2.2.5.1.3	correction to step 17: ~~Notify the client that the *Application* was instantiated successfully.~~Return the *Application* object reference.

Page	Section / Reference	Change
3-24	3.1.3.2.3.1	clarification to 5th para.: The administration operations are used to access the interfaces of registered *DeviceManagers* and *DomainManager's FileManagers*.
3-21	3.1.3.2.2.5.1.3	2rd para.: … The SPD specifies the *Device* implementation criteria for loading dependencies (processor kind, etc.) and processing capacities (e.g., memory, process) for an application component. …
3-27	3.1.3.2.3.6.1.3	clarification to 4th para.: The *registerDeviceManager* operation shall obtain all the Software profiles from the registering *DeviceManager's FileSystems*. Only installed applications that have been installed by the *DomainManager*::*InstallApplication* operation are used.
3-29	3.1.3.2.3.6.3.3	Correction in the third paragraph, The ~~*registerApplicaton*~~ *installApplication* operation …
3-36	3.1.3.2.4.5.2.3	clarification: The *execute* operation shall execute the given function with the input parameters. The parameters (IDs and format values) shall be: 1. prefix naming context – The ID is 1 and the value is a CORBA string. 2. stringified *DeviceManager* IOR – The ID is 2 and the value is a CORBA string.
3-39	3.1.3.2.4.5.9.3	clarification: The *executeProcess* operation shall execute the given fileName with the input parameters. If the input *FileSystem* is nil, then the *executeProcess* operation shall use the parent *Device's FileManager* for finding the file to be executed. The valid IDs and format values for parameters shall be: 1. prefix naming context – The ID is 1 and the value is a CORBA string. 2. stringified *DeviceManager* IOR – The ID is 2 and the value is CORBA string.
3-45	3.1.3.3.1.4.2	Correct *File* attribute text to match IDL: The FilePointer attribute provides read access to the file pointer position where the next read or write will occur. `Readonly attribute unsigned long filePointer;`

Page	Section / Reference	Change
3-56	3.1.3.3.4.1	Clarification: This interface is implemented by application developers for use by the *Logger* to push a string to consumers.
3-58	3.1.3.3.5.3	clarify use of *Logger* Types The logLevel type contains the valid levels for log data being logged and logs data being received. The logLevel shall be an unsigned short (16 bits) and is bitmapped 00 00 - 7F FF (hex). The MSB (d15) is a control bit to allow for log level manipulation. *Logger* Level manipulation using the control bit is as follows: 1. BIT ~~Matching~~ Mapping - When the control bit is one (1), setting / clearing the bit in the nth position affects the consumers and the producers log for level n only. ~~then l~~Logging or forwarding of log data shall be performed when the input log level of the log data matches registered consumer's or producer's log levels. Examples - LogLevel = C010 h (1100 0000 0001 0000~~b~~) indicates only levels 154 and 54 are to be sent to a consumer or logged for a producer. C0B3 h (1100 0000 1101 0011) indicates levels 15, 8, 7, 5, 2, and 1 are to be sent to a consumer or logged for a producer. 2. BIT Leveling - When the control bit is zero the level is interpreted by the MSB set of bits d14 – d0, where the MSB n indicates enable log reporting for levels n , n- 1, n-2, …, and 1. ~~then l~~Logging or forwarding log data shall be performed when the input log level is less than or equal to registered consumer or producer log level. Example - LogLevel = 000A h indicates levels 9 (10 least significant bits) and below will be sent to a consumer or logged for a producer, and bits 4-14 are unused. Examples: 0100 h (0000 0001 0000 0000) indicates levels 9 and below will be sent to a consumer or logged for a producer. 00E0 h (0000 0000 1110 0000) indicates levels 8 and below will be sent to a consumer or logged for a producer. The MSB set of d14 – d0 is d8 therefore d7 - d0 are not interpreted for registration of consumers or producers.

Page	Section / Reference	Change
		The logLevels can be set for consumers and producers. For producers the log level indicates the kind of information being logged. When using the logData method, the producer shall only set the appropriate log level of the log message. For consumers the log level determines what log information is sent to a consumer. The *Logger,* consumers, and producers shall use the log levels in the following table. The log levels in table 3-1 are listed in order of significance. Log level value 0x0001 is of the highest significance. Each increasing log level thereafter decreases in significance. An event may cause more than one (1) log message.
3-62	3.1.3.3.5.5.3.1	correction: Applications require the ~~*logData*~~ *setProducerLogLevel* operation in order to …
3-62	3.1.3.3.5.5.4.1	correction: Applications require the ~~*logData*~~ *setConsumerLogLevel* operation in order to …
3-69	3.2.1.3	clarification (2nd para.): … (In the naming parameter string, each "slash" (/) represents a separate naming context. The optional "[other context sequences]" allows for additional, implementation-unique context, e.g. node_process_ID.)
3-70 3-71	Figures 3-25 & 3-26	Figures are reversed (the figure identified as PushPort Data Interfaces contains the PullPort Data Interfaces and vice versa).